3.A.N. MICROPROCESSEURS
Devoir surveillé du 8 février 2002
Durée : 2 heures Documents autorisés

REMARQUE IMPORTANTE:

Les réponses mal écrites ou non commentées seront considérées comme fausses.

PARTIE A

La fiche technique jointe en annexe décrit les spécifications d'une fonction de librairie "bcopy()", utilisable dans un programme en langage "C" pour copier un bloc d'octets "source" vers un bloc "destination".

La description de la fonction évoque deux aspects de son fonctionnement :

- La possibilité que les blocs "source" et "destination" se chevauchent (ovelapping).
- Le mode de copie des données : par octets, words et ou long words, compte tenu de la taille du bloc à copier, mais aussi de l'alignement des adresses "source" et "destination", afin d'optimiser la vitesse d'exécution de la fonction.

QUESTION A-1

- Rédiger et commenter en **LANGAGE "ASM"** sur la fiche de programmation jointe, une première version du sous-programme **"bcopy_b"**, qui effectue la copie **PAR OCTETS,** sans gestion du chevauchement, et en respectant les spécifications suivantes :

Spécifications de la fonction "bcopy_b":

```
BUT :  copie d'un bloc d'octets "source" vers un bloc "destination"

P.E.:  A1.L = adresse du bloc "source".
       A2.L = adresse du bloc "destination".
       D1.W = nombre d'octets à copier.

P.S.:  Aucun.
```

QUESTION A-2

- Rédiger et commenter en **LANGAGE "ASM"** sur la fiche de programmation jointe, une seconde version du sous-programme "bcopy_b", qui effectue la copie par octets, sans gestion du chevauchement, mais de manière à le rendre utilisable en tant que **fonction "bcopy_b()"** dans un programme en langage "C" (donc en respectant la stratégie de programmation du compilateur "C").

QUESTION A-3

- Décrire sous forme algorithmique, la méthode à utiliser pour effectuer la copie, mais en supposant que les blocs "source" et "destination" se chevauchent partiellement (faire un dessin).
- Rédiger ET COMMENTER le texte source en **LANGAGE "C"** de la fonction " bcopyBytes()", en utilisant l'algorithme ci-dessus et en respectant la définition de la fiche technique correspondante.

QUESTION A-4

- Expliquer pourquoi la copie d'un word est plus rapide avec un µP 68000, que celle de 2 bytes consécutifs.
- Expliquer pourquoi la copie d'un long word est plus rapide avec un µP 68000, que celle de 2 words consécutifs.
- Décrire la méthode à utiliser dans chacun des 3 cas ci-dessous, pour effectuer la copie d'un bloc (sans chevauchement), par octets, words et ou long words, afin d'optimiser la vitesse d'exécution de la fonction "bcopy()".

cas	source	destination	nbytes
1	0xB000	0xC000	0x200
2	0xB000	0xC000	0x203
3	0xB000	0xC001	0x200

bcopy( )

NAME

bcopy( ) - copy one buffer to another

SYNOPSIS

```
void bcopy
    (
    const char  *source,       /* pointer to source buffer      */
    char        *destination,  /* pointer to destination buffer */
    int         nbytes         /* number of bytes to copy       */
    )
```

DESCRIPTION

This routine copies the first *nbytes* characters from *source* to *destination*. Overlapping buffers are handled correctly. Copying is done in the most efficient way possible, which may include long-word, or even multiple-long-word moves on some architectures. In general, the copy will be significantly faster if both buffers are long-word aligned. (For copying that is restricted to byte, word, or long-word moves, see the manual entries for *bcopyBytes*( ), *bcopyWords*( ), and *bcopyLongs*( ).)

RETURNS

N/A

SEE ALSO

bLib, *bcopyBytes*( ), *bcopyWords*( ), *bcopyLongs*( )

bcopyBytes( )

NAME

bcopyBytes( ) - copy one buffer to another one byte at a time

SYNOPSIS

```
void bcopyBytes
    (
    char  *source,       /* pointer to source buffer      */
    char  *destination,  /* pointer to destination buffer */
    int   nbytes         /* number of bytes to copy       */
    )
```

DESCRIPTION

This routine copies the first *nbytes* characters from *source* to *destination* one byte at a time. This may be desirable if a buffer can only be accessed with byte instructions, as in certain byte-wide memory-mapped peripherals.

RETURNS

N/A

SEE ALSO

FICHE DE PROGRAMMATION

	Référence	Opérande	Opcode	Commentaires
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				

PARTIE B

Le sous-programme "addxn" de la figure B-1 permet de calculer avec un µP 68000, la **SOMME MODULO N** de 2 octets binaires positifs P et Q (de valeurs inférieures à N).

On notera que N est paramétrable dans le registre D2.B, avec comme condition : **N <= 100**.

Le sous-programme "addxn" utilise en outre une macro-instruction "SETX" pour forcer à 1 le bit "X" du registre d'état du processeur.

QUESTION B-1

- Rédiger et commenter (sur la fiche de programmation) la **macro-instruction "SETX"** pour forcer à 1 le bit "X" du registre d'état, de manière à ne pas altérer les autres registres du processeur, ni les autres bits du registre d'état.
- Rédiger et commenter (sur la fiche de programmation) la **macro-instruction "CLRX"** pour forcer à 0 le bit "X" du registre d'état (dans les mêmes conditions).

QUESTION B-2

- Transformer le sous-programme "addxn" en **macro-instruction "ADDXN"** (sur la fiche de programmation), de manière à respecter les spécifications suivantes :

Spécifications de la macro-instruction " ADDXN ":

```
BUT :  addition modulo N d'octets binaires

P.E.:  \1 = valeur de N.
       \2 = adresse (ou valeur) du 1er octet à additionner.
       \3 = adresse (ou valeur) du 2ème octet à additionner.
       \4 = adresse de stockage du résultat.

P.S.:  X(SR) = report sortant de l'addition modulo N.
```

QUESTION B-3

- Rédiger et commenter un **programme testeur** pour la macro-instruction " ADDXN", en prenant N = 24, et de manière à vérifier les cas les plus intéressants.
- Décrire clairement la **procédure de test** en indiquant la séquence de commandes du moniteur "pRobe" à utiliser.

Figure B-1:

```
*=============================================================
*       ADDITION (MODULO N) D'OCTETS
*       P.E.:   d0.b = P
*               d1.b = Q
*               d2.b = N
*       P.S.:   d0.b = (P + Q + x) % N
*                  X = report sortant de l'addition modulo N
*=============================================================
addxn   addx.b  d1,d0
        cmp.b   d2,d0
        bcs     addxn1
        sub.b   d2,d0
        SETX
addxn1  rts
*=============================================================
```

FICHE DE PROGRAMMATION

	Référence	Opérande	Opcode	Commentaires
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				